

# Chapter B15. Modeling of Data

```
SUBROUTINE fit(x,y,a,b,siga,sigb,chi2,q,sig)
USE nrtype; USE nrutil, ONLY : assert_eq
USE nr, ONLY : gammq
IMPLICIT NONE
REAL(SP), DIMENSION(:, ), INTENT(IN) :: x,y
REAL(SP), INTENT(OUT) :: a,b,siga,sigb,chi2,q
REAL(SP), DIMENSION(:, ), OPTIONAL, INTENT(IN) :: sig
Given a set of data points in same-size arrays x and y, fit them to a straight line  $y = a + bx$ 
by minimizing  $\chi^2$ . sig is an optional array of the same length containing the individual
standard deviations. If it is present, then a,b are returned with their respective probable
uncertainties siga and sigb, the chi-square chi2, and the goodness-of-fit probability q
(that the fit would have  $\chi^2$  this large or larger). If sig is not present, then q is returned
as 1.0 and the normalization of chi2 is to unit standard deviation on all points.
INTEGER(I4B) :: ndata
REAL(SP) :: sigdat,ss,sx,sxoss,sy,st2
REAL(SP), DIMENSION(size(x)), TARGET :: t
REAL(SP), DIMENSION(:, ), POINTER :: wt
if (present(sig)) then
    ndata=assert_eq(size(x),size(y),size(sig),'fit')
    wt=>t                                     Use temporary variable t to store weights.
    wt(:)=1.0_sp/(sig(:)**2)
    ss=sum(wt(:))                                Accumulate sums with weights.
    sx=dot_product(wt,x)
    sy=dot_product(wt,y)
else
    ndata=assert_eq(size(x),size(y),'fit')
    ss=real(size(x),sp)                         Accumulate sums without weights.
    sx=sum(x)
    sy=sum(y)
end if
sxoss=sx/ss
t(:)=x(:)-sxoss
if (present(sig)) then
    t(:)=t(:)/sig(:)
    b=dot_product(t/sig,y)
else
    b=dot_product(t,y)
end if
st2=dot_product(t,t)
b=b/st2                                         Solve for a, b,  $\sigma_a$ , and  $\sigma_b$ .
a=(sy-sx*b)/ss
siga=sqrt((1.0_sp+sx*sx/(ss*st2))/ss)
sigb=sqrt(1.0_sp/st2)
t(:)=y(:)-a-b*x(:)
q=1.0
if (present(sig)) then
    t(:)=t(:)/sig(:)
    chi2=dot_product(t,t)                      Calculate  $\chi^2$ .
    if (ndata > 2) q=gammq(0.5_sp*(size(x)-2),0.5_sp*chi2)   Equation (15.2.12).
else
    chi2=dot_product(t,t)
```

Sample page from NUMERICAL RECIPES IN FORTRAN 90: THE Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 by Numerical Recipes Software.  
Permission is granted for internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-  
readable files (including this one) to any server computer, is strictly prohibited. To order Numerical Recipes books or CDROMs, visit website  
<http://www.nr.com> or call 1-800-872-7423 (North America only), or send email to [directcuiserv@cambridge.org](mailto:directcuiserv@cambridge.org) (outside North America).

```

sigdat=sqrt(chi2/(size(x)-2))
siga=siga*sigdat
sigb=sigb*sigdat
end if
END SUBROUTINE fit

```

For unweighted data evaluate typical sig using chi2, and adjust the standard deviations.

**f90** REAL(SP), DIMENSION(:), POINTER :: wt...wt=>t When standard deviations are supplied in sig, we need to compute the weights for the least squares fit in a temporary array wt. Later in the routine, we need another temporary array, which we call t to correspond to the variable in equation (15.2.15). It would be confusing to use the same name for both arrays. In Fortran 77 the arrays could share storage with an EQUIVALENCE declaration, but that is a deprecated feature in Fortran 90. We accomplish the same thing by making wt a pointer alias to t.

\* \* \*

```

SUBROUTINE fitexy(x,y,sigx,sigy,a,b,siga,sigb,chi2,q)
USE nrtype; USE nrutil, ONLY : assert_eq,swap
USE nr, ONLY : avevar,brent,fit,gammq,mnbrak,zbrent
USE chixyfit
IMPLICIT NONE
REAL(SP), DIMENSION(:), INTENT(IN) :: x,y,sigx,sigy
REAL(SP), INTENT(OUT) :: a,b,siga,sigb,chi2,q
REAL(SP), PARAMETER :: POTN=1.571000_sp,BIG=1.0e30_sp,ACC=1.0e-3_sp
Straight-line fit to input data x and y with errors in both x and y, the respective standard deviations being the input quantities sigx and sigy. x, y, sigx, and sigy are all arrays of the same length. Output quantities are a and b such that  $y = a + bx$  minimizes  $\chi^2$ , whose value is returned as chi2. The  $\chi^2$  probability is returned as q, a small value indicating a poor fit (sometimes indicating underestimated errors). Standard errors on a and b are returned as siga and sigb. These are not meaningful if either (i) the fit is poor, or (ii) b is so large that the data are consistent with a vertical (infinite b) line. If siga and sigb are returned as BIG, then the data are consistent with all values of b.
INTEGER(I4B) :: j,n
REAL(SP), DIMENSION(size(x)), TARGET :: xx,yy,sx,sy,ww
REAL(SP), DIMENSION(6) :: ang,ch
REAL(SP) :: amx,amn,varx,vary,scale,bmn,bmx,d1,d2,r2,&
           dum1,dum2,dum3,dum4,dum5
n=assert_eq(size(x),size(y),size(sigx),size(sigy),'fitexy')
xxp=>xx
yyp=>yy
sxp=>sx
syp=>sy
wwp=>ww
call avevar(x,dum1,varx)          Find the x and y variances, and scale the
call avevar(y,dum1,vary)          data.
scale=sqrt(varx/vary)
xx(:)=x(:)
yy(:)=y(:)*scale
sx(:)=sigx(:)
sy(:)=sigy(:)*scale
ww(:)=sqrt(sx(:)**2+sy(:)**2)      Use both x and y weights in first trial fit.
call fit(xx,yy,dum1,b,dum2,dum3,dum4,dum5,ww)    Trial fit for b.
offs=0.0
ang(1)=0.0
ang(2)=atan(b)
ang(4)=0.0
ang(5)=ang(2)
ang(6)=POTN
do j=4,6
  ch(j)=chixy(ang(j))

```

Sample page from NUMERICAL RECIPES IN FORTRAN 90: THE Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 by Numerical Recipes Software.  
Permission is granted for Internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-readable files (including this one), to any server computer, is strictly prohibited. To order Numerical Recipes books or CDROMs, visit website <http://www.nr.com> or call 1-800-872-7423 (North America only), or send email to [directcustserv@cambridge.org](mailto:directcustserv@cambridge.org) (outside North America).

```

end do
call mnbrak(ang(1),ang(2),ang(3),ch(1),ch(2),ch(3),chixy)
    Bracket the  $\chi^2$  minimum and then locate it with brent.
chi2=brent(ang(1),ang(2),ang(3),chixy,ACC,b)
chi2=chixy(b)
a=aa
q=gammq(0.5_sp*(n-2),0.5_sp*chi2)           Compute  $\chi^2$  probability.
r2=1.0_sp/sum(ww(:))                         Save inverse sum of weights at the minimum.
bmx=BIG                                         Now, find standard errors for  $b$  as points where
bmn=BIG                                          $\Delta\chi^2 = 1$ .
offs=chi2+1.0_sp
do j=1,6
    if (ch(j) > offs) then
        d1=mod(abs(ang(j)-b),PI)
        d2=PI-d1
        if (ang(j) < b) call swap(d1,d2)
        if (d1 < bmx) bmx=d1
        if (d2 < bmn) bmn=d2
    end if
end do
if (bmx < BIG) then                          Call zbrent to find the roots.
    bmx=zbrent(chixy,b,b+bmx,ACC)-b
    amx=aa-a
    bmn=zbrent(chixy,b,b-bmn,ACC)-b
    amn=aa-a
    sigb=sqrt(0.5_sp*(bmx**2+bm**2))/(scale*cos(b)**2)
    siga=sqrt(0.5_sp*(amx**2+amn**2)+r2)/scale      Error in a has additional piece
else                                           r2.
    sigb=BIG
    siga=BIG
end if
a=a/scale                                     Unscale the answers.
b=tan(b)/scale
END SUBROUTINE fitexy

```

**f90** USE chixyfit We need to pass arrays and other variables to chixy, but not as arguments. See §21.5 and the discussion of fminln on p. 1197 for two good ways to do this. The pointer construction here is analogous to the one used in fminln.

```

MODULE chixyfit
USE nrtype; USE nrutil, ONLY : nrerror
REAL(SP), DIMENSION(:, ), POINTER :: xxp,yyp,sxp,syp,wwp
REAL(SP) :: aa,offs
CONTAINS
FUNCTION chixy(bang)
IMPLICIT NONE
REAL(SP), INTENT(IN) :: bang
REAL(SP) :: chixy
REAL(SP), PARAMETER :: BIG=1.0e30_sp
    Captive function of fitexy, returns the value of  $(\chi^2 - \text{offs})$  for the slope  $b=\tan(\text{bang})$ .
    Scaled data and offs are communicated via the module chixyfit.
REAL(SP) :: avex,avey,sumw,b
if (.not. associated(wwp)) call nrerror("chixy: bad pointers")
b=tan(bang)
wwp(:)=(b*sxp(:))**2+syp(:)**2
where (wwp(:) < 1.0/BIG)
    wwp(:)=BIG
elsewhere
    wwp(:)=1.0_sp/wwp(:)
end where

```

Sample page from NUMERICAL RECIPES IN FORTRAN 90: THE Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 by Numerical Recipes Software.  
Permission is granted for Internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-readable files (including this one), to any server computer, is strictly prohibited. To order Numerical Recipes books or CDROMs, visit website <http://www.nr.com> or call 1-800-872-7423 (North America only), or send email to [directcustserv@cambridge.org](mailto:directcustserv@cambridge.org) (outside North America).

```

sumw=sum(wwp)
avex=dot_product(wwp,xxp)/sumw
avey=dot_product(wwp,ypy)/sumw
aa=avey-b*avex
chixy=sum(wwp(:)*(ypy(:)-aa-b*xxp(:))**2)-offs
END FUNCTION chixy
END MODULE chixyfit

```

★      ★      ★

```

SUBROUTINE lfit(x,y,sig,a,maska,covar,chisq,funcs)
USE nrtype; USE nrutil, ONLY : assert_eq,diagmult,nrerror
USE nr, ONLY : covsrt,gaussj
IMPLICIT NONE
REAL(SP), DIMENSION(:, ), INTENT(IN) :: x,y,sig
REAL(SP), DIMENSION(:, ), INTENT(INOUT) :: a
LOGICAL(LGT), DIMENSION(:, ), INTENT(IN) :: maska
REAL(SP), DIMENSION(:, :, ), INTENT(INOUT) :: covar
REAL(SP), INTENT(OUT) :: chisq
INTERFACE
    SUBROUTINE funcs(x,arr)
        USE nrtype
        IMPLICIT NONE
        REAL(SP),INTENT(IN) :: x
        REAL(SP), DIMENSION(:, ), INTENT(OUT) :: arr
    END SUBROUTINE funcs
END INTERFACE

```

Given a set of  $N$  data points  $x, y$  with individual standard deviations  $\text{sig}$ , all arrays of length  $N$ , use  $\chi^2$  minimization to fit for some or all of the  $M$  coefficients  $a$  of a function that depends linearly on  $a$ ,  $y = \sum_{i=1}^M a_i \times \text{afunc}_i(x)$ . The input logical array  $\text{maska}$  of length  $M$  indicates by true entries those components of  $a$  that should be fitted for, and by false entries those components that should be held fixed at their input values. The program returns values for  $a$ ,  $\chi^2 = \text{chisq}$ , and the  $M \times M$  covariance matrix  $\text{covar}$ . (Parameters held fixed will return zero covariances.) The user supplies a subroutine  $\text{funcs}(x, \text{afunc})$  that returns the  $M$  basis functions evaluated at  $x = \mathbf{x}$  in the array  $\text{afunc}$ .

```

INTEGER(I4B) :: i,j,k,l,ma,mfit,n
REAL(SP) :: sig2i,wt,ym
REAL(SP), DIMENSION(size(maska)) :: afunc
REAL(SP), DIMENSION(size(maska),1) :: beta
n(assert_eq(size(x),size(y),size(sig),'lfit: n'))
ma(assert_eq(size(maska),size(a),size(covar,1),size(covar,2),'lfit: ma'))
mfit=count(maska)                                     Number of parameters to fit for.
if (mfit == 0) call nrerror('lfit: no parameters to be fitted')
covar(1:mfit,1:mfit)=0.0                             Initialize the (symmetric) matrix.
beta(1:mfit,1)=0.0
do i=1,n                                         Loop over data to accumulate coefficients of
    call funcs(x(i),afunc)                         the normal equations.
    ym=y(i)
    if (mfit < ma) ym=ym-sum(a(1:ma)*afunc(1:ma), mask=.not. maska)
        Subtract off dependences on known pieces of the fitting function.
    sig2i=1.0_sp/sig(i)**2
    j=0
    do l=1,ma
        if (maska(l)) then
            j=j+1
            wt=afunc(l)*sig2i
            k=count(maska(1:l))
            covar(j,1:k)=covar(j,1:k)+wt*pack(afunc(1:l),maska(1:l))
            beta(j,1)=beta(j,1)+ym*wt
        end if
    end do

```

Sample page from NUMERICAL RECIPES IN FORTRAN 90: THE Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)

Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 by Numerical Recipes Software.

Permission is granted for Internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-readable files (including this one), to any server computer, is strictly prohibited. To order Numerical Recipes books or CDROMs, visit website <http://www.nr.com> or call 1-800-872-7423 (North America only), or send email to [directcustserv@cambridge.org](mailto:directcustserv@cambridge.org) (outside North America).

```

end do
call diagmult(covar(1:mfit,1:mfit),0.5_sp)
covar(1:mfit,1:mfit)= &           Fill in above the diagonal from symmetry.
    covar(1:mfit,1:mfit)+transpose(covar(1:mfit,1:mfit))
call gaussj(covar(1:mfit,1:mfit),beta(1:mfit,1:1))      Matrix solution.
a(1:ma)=unpack(beta(1:ma,1),maska,a(1:ma))
Partition solution to appropriate coefficients a.
chisq=0.0          Evaluate  $\chi^2$  of the fit.
do i=1,n
    call func(x(i),afunc)
    chisq=chisq+((y(i)-dot_product(a(1:ma),afunc(1:ma)))/sig(i))**2
end do
call covsrt(covar,maska)          Sort covariance matrix to true order of fitting
END SUBROUTINE lfit               coefficients.

```

**f90**

if (mfit < ma) ym=ym-sum(a(1:ma)\*afunc(1:ma), mask=.not. maska)  
 This is the first of several uses of maska in this routine to control  
 which elements of an array are to be used. Here we include in the sum  
 only elements for which maska is false, i.e., elements corresponding to parameters  
 that are not being fitted for.

covar(j,1:k)=covar(j,1:k)+wt\*pack(afunc(1:l),maska(1:l)) Here maska  
 controls which elements of afunc get packed into the covariance matrix.

call diagmult(covar(1:mfit,1:mfit),0.5\_sp) See discussion of diagadd  
 after hqr on p. 1234.

a(1:ma)=unpack(beta(1:ma,1),maska,a(1:ma)) And here maska controls which  
 elements of beta get unpacked into the appropriate slots in a. Where maska is  
 false, corresponding elements are selected from the third argument of unpack, here  
 a itself. The net effect is that those elements remain unchanged.

★      ★      ★

```

SUBROUTINE covsrt(covar,maska)
USE nrtype; USE nrutil, ONLY : assert_eq,swap
IMPLICIT NONE
REAL(SP), DIMENSION(:,:), INTENT(INOUT) :: covar
LOGICAL(LGT), DIMENSION(:), INTENT(IN) :: maska
  Expand in storage the covariance matrix covar, so as to take into account parameters that
  are being held fixed. (For the latter, return zero covariances.)
INTEGER(I4B) :: ma,mfit,j,k
ma=assert_eq(size(covar,1),size(covar,2),size(maska),'covsrt')
mfit=count(maska)
covar(mfit+1:ma,1:ma)=0.0
covar(1:ma,mfit+1:ma)=0.0
k=mfit
do j=ma,1,-1
    if (maska(j)) then
        call swap(covar(1:ma,k),covar(1:ma,j))
        call swap(covar(k,1:ma),covar(j,1:ma))
        k=k-1
    end if
end do
END SUBROUTINE covsrt

```

★      ★      ★

Sample page from NUMERICAL RECIPES IN FORTRAN 90: THE Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
 Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 by Numerical Recipes Software.  
 Permission is granted for Internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-  
 readable files (including this one), to any server computer, is strictly prohibited. To order Numerical Recipes books or CDROMs, visit website  
<http://www.nr.com> or call 1-800-872-7423 (North America only), or send email to [directcustserv@cambridge.org](mailto:directcustserv@cambridge.org) (outside North America).

```

SUBROUTINE svdfit(x,y,sig,a,v,w,chisq,funcs)
USE nrtype; USE nrutil, ONLY : assert_eq,vabs
USE nr, ONLY : svbksb,svdcmp
IMPLICIT NONE
REAL(SP), DIMENSION(:,), INTENT(IN) :: x,y,sig
REAL(SP), DIMENSION(:,), INTENT(OUT) :: a,w
REAL(SP), DIMENSION(:,:), INTENT(OUT) :: v
REAL(SP), INTENT(OUT) :: chisq
INTERFACE
  FUNCTION funcs(x,n)
    USE nrtype
    IMPLICIT NONE
    REAL(SP), INTENT(IN) :: x
    INTEGER(I4B), INTENT(IN) :: n
    REAL(SP), DIMENSION(n) :: funcs
  END FUNCTION funcs
END INTERFACE
REAL(SP), PARAMETER :: TOL=1.0e-5_sp
Given a set of  $N$  data points  $x, y$  with individual standard deviations  $\text{sig}$ , all arrays of length  $N$ , use  $\chi^2$  minimization to determine the  $M$  coefficients  $a$  of a function that depends linearly on  $a$ ,  $y = \sum_{i=1}^M a_i \times \text{afunc}_i(x)$ . Here we solve the fitting equations using singular value decomposition of the  $N \times M$  matrix, as in §2.6. On output, the  $M \times M$  array  $v$  and the vector  $w$  of length  $M$  define part of the singular value decomposition, and can be used to obtain the covariance matrix. The program returns values for the  $M$  fit parameters  $a$ , and  $\chi^2$ ,  $\text{chisq}$ . The user supplies a function  $\text{funcs}(x, \text{afunc})$  that returns the  $M$  basis functions evaluated at  $x = X$  in the array  $\text{afunc}$ .
Parameter: Above value of  $\text{TOL}$  is for single precision and variables scaled to order unity.
INTEGER(I4B) :: i,ma,n
REAL(SP), DIMENSION(size(x)) :: b,sigi
REAL(SP), DIMENSION(size(x),size(a)) :: u,usav
n=assert_eq(size(x),size(y),size(sig),'svdfit: n')
ma=assert_eq(size(a),size(v,1),size(v,2),size(w),'svdfit: ma')
sigi=1.0_sp/sig
b=y*sigi
Accumulate coefficients of the fitting matrix in
u.
do i=1,n
  usav(i,:)=funcs(x(i),ma)
end do
u=usav*spread(sigi,dim=2,ncopies=ma)
usav=u
call svdcmp(u,v,v)
Singular value decomposition.
where (w < TOL*maxval(w)) w=0.0
Edit the singular values, given  $\text{TOL}$  from the pa-
call svbksb(u,w,v,b,a)
rameter statement.
chisq=vabs(matmul(usav,a)-b)**2
Evaluate chi-square.
END SUBROUTINE svdfit

```

**f90**  $u=usav*spread(sigi,dim=2,ncopies=ma)$  Remember how `spread` works: the vector `sigi` is copied *along* the dimension 2, making a matrix whose columns are each a copy of `sigi`. The multiplication here is element by element, so each row of `usav` is multiplied by the corresponding element of `sigi`.

$\text{chisq}=vabs(\text{matmul}(\text{usav},\text{a})-\text{b})^{**2}$  Fortran 90's `matmul` intrinsic allows us to evaluate  $\chi^2$  from the mathematical definition in terms of matrices. `vabs` in `nrutil` returns the length of a vector ( $L_2$  norm).

```

SUBROUTINE svdvar(v,w,cvm)
USE nrtype; USE nrutil, ONLY : assert_eq
IMPLICIT NONE
REAL(SP), DIMENSION(:,:), INTENT(IN) :: v
REAL(SP), DIMENSION(:), INTENT(IN) :: w
REAL(SP), DIMENSION(:,:), INTENT(OUT) :: cvm

```

Sample page from NUMERICAL RECIPES IN FORTRAN 90: THE Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 by Numerical Recipes Software.  
Permission is granted for Internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-readable files (including this one), to any server computer, is strictly prohibited. To order Numerical Recipes books or CDROMs, visit website  
<http://www.nr.com> or call 1-800-872-7423 (North America only), or send email to [directcustserv@cambridge.org](mailto:directcustserv@cambridge.org) (outside North America).

To evaluate the covariance matrix `cvm` of the fit for  $M$  parameters obtained by `svdfit`, call this routine with matrices `v,w` as returned from `svdfit`. The dimensions are  $M$  for `w` and  $M \times M$  for `v` and `cvm`.

```
INTEGER(I4B) :: ma
REAL(SP), DIMENSION(size(w)) :: wti
ma=assert_eq((/size(v,1),size(v,2),size(w),size(cvm,1),size(cvm,2)/),&
             'svdvar')
where (w /= 0.0)
    wti=1.0_sp/(w*w)
elsewhere
    wti=0.0
end where
cvm=v*spread(wti,dim=1,ncopies=ma)
cvm=matmul(cvm,transpose(v))          Covariance matrix is given by (15.4.20).
END SUBROUTINE svdvar
```

**f90**

where (`w /= 0.0`)...`elsewhere`...`end where` This is the standard Fortran 90 construction for doing different things to a matrix depending on some condition. Here we want to avoid inverting elements of `w` that are zero.

`cvm=v*spread(wti,dim=1,ncopies=ma)` Each column of `v` gets multiplied by the corresponding element of `wti`. Contrast the construction `spread(...dim=2...)` in `svdfit`.

★      ★      ★

```
FUNCTION fpoly(x,n)
USE nrtype; USE nrutil, ONLY : geop
IMPLICIT NONE
REAL(SP), INTENT(IN) :: x
INTEGER(I4B), INTENT(IN) :: n
REAL(SP), DIMENSION(n) :: fpoly
    Fitting routine for a polynomial of degree  $n - 1$ , returning  $n$  coefficients in fpoly.
fpoly=geop(1.0_sp,x,n)
END FUNCTION fpoly
```

★      ★      ★

```
FUNCTION fleg(x,nl)
USE nrtype
IMPLICIT NONE
REAL(SP), INTENT(IN) :: x
INTEGER(I4B), INTENT(IN) :: nl
REAL(SP), DIMENSION(nl) :: fleg
    Fitting routine for an expansion with  $nl$  Legendre polynomials evaluated at x and returned
    in the array fleg of length nl. The evaluation uses the recurrence relation as in §5.5.
INTEGER(I4B) :: j
REAL(SP) :: d,f1,f2,twox
fleg(1)=1.0
fleg(2)=x
if (nl > 2) then
    twox=2.0_sp*x
    f2=x
    d=1.0
    do j=3, nl
        f1=d
        f2=f2+twox
        d=d+1.0_sp
```

Sample page from NUMERICAL RECIPES IN FORTRAN 90: THE Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 by Numerical Recipes Software.  
Permission is granted for Internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-readable files (including this one), to any server computer, is strictly prohibited. To order Numerical Recipes books or CDROMs, visit website <http://www.nr.com> or call 1-800-872-7423 (North America only), or send email to [directcustserv@cambridge.org](mailto:directcustserv@cambridge.org) (outside North America).

```

    fleg(j)=(f2*fleg(j-1)-f1*fleg(j-2))/d
end do
end if
END FUNCTION fleg

```

\* \* \*

```

SUBROUTINE mrqmin(x,y,sig,a,maska,covar,alpha,chisq,funcs,alamda)
USE nrtype; USE nrutil, ONLY : assert_eq,diagmult
USE nr, ONLY : covsrt,gaussj
IMPLICIT NONE
REAL(SP), DIMENSION(:), INTENT(IN) :: x,y,sig
REAL(SP), DIMENSION(:), INTENT(INOUT) :: a
REAL(SP), DIMENSION(:, :, ), INTENT(OUT) :: covar,alpha
REAL(SP), INTENT(OUT) :: chisq
REAL(SP), INTENT(INOUT) :: alamda
LOGICAL(LGT), DIMENSION(:), INTENT(IN) :: maska
INTERFACE
    SUBROUTINE funcs(x,a,yfit,dyda)
    USE nrtype
    REAL(SP), DIMENSION(:), INTENT(IN) :: x,a
    REAL(SP), DIMENSION(:, ), INTENT(OUT) :: yfit
    REAL(SP), DIMENSION(:, :, ), INTENT(OUT) :: dyda
    END SUBROUTINE funcs
END INTERFACE
Levenberg-Marquardt method, attempting to reduce the value  $\chi^2$  of a fit between a set of  $N$  data points  $x$ ,  $y$  with individual standard deviations  $sig$ , and a nonlinear function dependent on  $M$  coefficients  $a$ . The input logical array  $maska$  of length  $M$  indicates by true entries those components of  $a$  that should be fitted for, and by false entries those components that should be held fixed at their input values. The program returns current best-fit values for the parameters  $a$ , and  $\chi^2 = chisq$ . The  $M \times M$  arrays  $covar$  and  $alpha$  are used as working space during most iterations. Supply a subroutine  $funcs(x,a,yfit,dyda)$  that evaluates the fitting function  $yfit$ , and its derivatives  $dyda$  with respect to the fitting parameters  $a$  at  $x$ . On the first call provide an initial guess for the parameters  $a$ , and set  $alamda < 0$  for initialization (which then sets  $alamda=.001$ ). If a step succeeds  $chisq$  becomes smaller and  $alamda$  decreases by a factor of 10. If a step fails  $alamda$  grows by a factor of 10. You must call this routine repeatedly until convergence is achieved. Then, make one final call with  $alamda=0$ , so that  $covar$  returns the covariance matrix, and  $alpha$  the curvature matrix. (Parameters held fixed will return zero covariances.)
INTEGER(I4B) :: ma,ndata
INTEGER(I4B), SAVE :: mfit
call mrqmin_private
CONTAINS
SUBROUTINE mrqmin_private
REAL(SP), SAVE :: ochisq
REAL(SP), DIMENSION(:), ALLOCATABLE, SAVE :: atryp,beta
REAL(SP), DIMENSION(:, :, ), ALLOCATABLE, SAVE :: da
ndata=assert_eq(size(x),size(y),size(sig),'mrqmin: ndata')
ma=assert_eq(/size(a),size(maska),size(covar,1),size(covar,2),&
    size(alpha,1),size(alpha,2)/,'mrqmin: ma')
mfit=count(maska)
if (alamda < 0.0) then
    allocate(atry(ma),beta(ma),da(ma,1)) Initialization.
    alamda=0.001_sp
    call mqcof(a,alpha,beta)
    ochisq=chisq
    atryp=a
end if
covar(1:mfit,1:mfit)=alpha(1:mfit,1:mfit)
call diagmult(covar(1:mfit,1:mfit),1.0_sp+alamda)

```

Sample page from NUMERICAL RECIPES IN FORTRAN 90: THE Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 by Numerical Recipes Software.  
Permission is granted for Internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-readable files (including this one), to any server computer, is strictly prohibited. To order Numerical Recipes books or CDROMs, visit website <http://www.nr.com> or call 1-800-872-7423 (North America only), or send email to [directcustserv@cambridge.org](mailto:directcustserv@cambridge.org) (outside North America).

```

        Alter linearized fitting matrix, by augmenting diagonal elements.
da(1:mfit,1)=beta(1:mfit)
call gaussj(covar(1:mfit,1:mfit),da(1:mfit,1:1))      Matrix solution.
if (alamda == 0.0) then                                Once converged, evaluate covariance ma-
    call covsrt(covar,maska)                           trix.
    call covsrt(alpha,maska)                          Spread out alpha to its full size too.
    deallocate(atry,beta,da)
    RETURN
end if
atry=a+unpack(da(1:mfit,1),maska,0.0_sp)      Did the trial succeed?
call mrqcof(atry,covar,da(1:mfit,1))
if (chisq < ochisq) then                         Success, accept the new solution.
    alamda=0.1_sp*alamda
    ochisq=chisq
    alpha(1:mfit,1:mfit)=covar(1:mfit,1:mfit)
    beta(1:mfit)=da(1:mfit,1)
    a=atry
else                                              Failure, increase alamda and return.
    alamda=10.0_sp*alamda
    chisq=ochisq
end if
END SUBROUTINE mrqmin_private

SUBROUTINE mrqcof(a,alpha,beta)
REAL(SP), DIMENSION(:), INTENT(IN) :: a
REAL(SP), DIMENSION(:), INTENT(OUT) :: beta
REAL(SP), DIMENSION(:,:), INTENT(OUT) :: alpha
Used by mrqmin to evaluate the linearized fitting matrix alpha, and vector beta as in
(15.5.8), and calculate  $\chi^2$ .
INTEGER(I4B) :: j,k,l,m
REAL(SP), DIMENSION(size(x),size(a)) :: dyda
REAL(SP), DIMENSION(size(x)) :: dy,sig2i,wt,ymod
call funcs(x,a,ymod,dyda)                      Loop over all the data.
sig2i=1.0_sp/(sig**2)
dy=y-ymod
j=0
do l=1,ma
    if (maska(l)) then
        j=j+1
        wt=dyda(:,l)*sig2i
        k=0
        do m=1,l
            if (maska(m)) then
                k=k+1
                alpha(j,k)=dot_product(wt,dyda(:,m))
                alpha(k,j)=alpha(j,k)          Fill in the symmetric side.
            end if
        end do
        beta(j)=dot_product(dy,wt)
    end if
end do
chisq=dot_product(dy**2,sig2i)                  Find  $\chi^2$ .
END SUBROUTINE mrqcof
END SUBROUTINE mrqmin

```

**f90** The organization of this routine is similar to that of amoeba, discussed on p. 1209. We want to keep the argument list of mrqcof to a minimum, but we want to make clear what global variables it accesses, and protect mrqmin\_private's name space.

REAL(SP), DIMENSION(:), ALLOCATABLE, SAVE :: atry,beta These arrays, as well as da, are allocated with the correct dimensions on the first call to mrqmin.

Sample page from NUMERICAL RECIPES IN FORTRAN 90: THE Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 by Numerical Recipes Software.  
Permission is granted for Internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-readable files (including this one), to any server computer, is strictly prohibited. To order Numerical Recipes books or CDROMs, visit website <http://www.nr.com> or call 1-800-872-7423 (North America only), or send email to [directcustserv@cambridge.org](mailto:directcustserv@cambridge.org) (outside North America).

They need to retain their values between calls, so they are declared with the SAVE attribute. They get deallocated only on the final call when alamda=0.

call diagmult(...) See discussion of diagadd after hqr on p. 1234.

atry=a+unpack(da(1:mfit,1),maska,0.0\_sp) maska controls which elements of a get incremented by da and which by 0.

★ ★ ★

```
SUBROUTINE fgauss(x,a,y,dyda)
USE nrtype; USE nrutil, ONLY : assert_eq
IMPLICIT NONE
REAL(SP), DIMENSION(:, ), INTENT(IN) :: x,a
REAL(SP), DIMENSION(:, ), INTENT(OUT) :: y
REAL(SP), DIMENSION(:, :, ), INTENT(OUT) :: dyda
y(x;a) is the sum of N/3 Gaussians (15.5.16). Here N is the length of the vectors x, y
and a, while dyda is an N × N matrix. The amplitude, center, and width of the Gaussians
are stored in consecutive locations of a: a(i) = Bk, a(i+1) = Ek, a(i+2) = Gk,
k = 1, ..., N/3.
INTEGER(I4B) :: i,na,nx
REAL(SP), DIMENSION(size(x)) :: arg,ex,fac
nx=assert_eq(size(x),size(y),size(dyda,1),'fgauss: nx')
na=assert_eq(size(a),size(dyda,2),'fgauss: na')
y(:)=0.0
do i=1,na-1,3
    arg(:)=(x(:)-a(i+1))/a(i+2)
    ex(:)=exp(-arg(:)**2)
    fac(:)=a(i)*ex(:)*2.0_sp*arg(:)
    y(:)=y(:)+a(i)*ex(:)
    dyda(:,i)=ex(:)
    dyda(:,i+1)=fac(:)/a(i+2)
    dyda(:,i+2)=fac(:)*arg(:)/a(i+2)
end do
END SUBROUTINE fgauss
```

★ ★ ★

```
SUBROUTINE medfit(x,y,a,b,abdev)
USE nrtype; USE nrutil, ONLY : assert_eq
USE nr, ONLY : select
IMPLICIT NONE
REAL(SP), DIMENSION(:, ), INTENT(IN) :: x,y
REAL(SP), INTENT(OUT) :: a,b,abdev
Fits y = a + bx by the criterion of least absolute deviations. The same-size arrays x and y are
the input experimental points. The fitted parameters a and b are output, along with abdev,
which is the mean absolute deviation (in y) of the experimental points from the fitted line.
INTEGER(I4B) :: ndata
REAL(SP) :: aa
call medfit_private
CONTAINS
SUBROUTINE medfit_private
IMPLICIT NONE
REAL(SP) :: b1,b2,bb,chisq,del,f,f1,f2,sigb,sx,sxx,sxy,sy
REAL(SP), DIMENSION(size(x)) :: tmp
nndata=assert_eq(size(x),size(y),'medfit')
sx=sum(x) As a first guess for a and b, we will find the least
sy=sum(y) squares fitting line.
sxy=dot_product(x,y)
```

Sample page from NUMERICAL RECIPES IN FORTRAN 90: THE Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 by Numerical Recipes Software.  
Permission is granted for Internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-readable files (including this one), to any server computer, is strictly prohibited. To order Numerical Recipes books or CDROMs, visit website  
<http://www.nr.com> or call 1-800-872-7423 (North America only), or send email to [directcustserv@cambridge.org](mailto:directcustserv@cambridge.org) (outside North America).

```

sxx=dot_product(x,x)
del=ndata*sxx-sx**2
aa=(sxx*sy-sx*sxy)/del
bb=(ndata*sxy-sx*sxy)/del
tmp(:)=y(:)-(aa+bb*x(:))
chisq=dot_product(tmp,tmp)
sigb=sqrt(chisq/del)
b1=bb
f1=rofunc(b1)
if (sigb > 0.0) then
  b2=bb+sign(3.0_sp*sigb,f1)
  f2=rofunc(b2)
  if (b2 == b1) then
    a=aa
    b=bb
    RETURN
  endif
  do
    if (f1*f2 <= 0.0) exit
    bb=b2+1.6_sp*(b2-b1)
    b1=b2
    f1=f2
    b2=bb
    f2=rofunc(b2)
  end do
  sigb=0.01_sp*sigb
  do
    if (abs(b2-b1) <= sigb) exit
    bb=b1+0.5_sp*(b2-b1)           Bisection.
    if (bb == b1 .or. bb == b2) exit
    f=rofunc(bb)
    if (f*f1 >= 0.0) then
      f1=f
      b1=bb
    else
      f2=f
      b2=bb
    end if
  end do
end if
a=aa
b=bb
abdev=abdev/ndata
END SUBROUTINE medfit_private

FUNCTION rofunc(b)
IMPLICIT NONE
REAL(SP), INTENT(IN) :: b
REAL(SP) :: rofunc
REAL(SP), PARAMETER :: EPS=epsilon(b)
Evaluates the right-hand side of equation (15.7.16) for a given value of b.
INTEGER(I4B) :: j
REAL(SP), DIMENSION(size(x)) :: arr,d
arr(:)=y(:)-b*x(:)
if (mod(ndata,2) == 0) then
  j=ndata/2
  aa=0.5_sp*(select(j,arr)+select(j+1,arr))
else
  aa=select((ndata+1)/2,arr)
end if
d(:)=y(:)-(b*x(:)+aa)
abdev=sum(abs(d))
where (y(:) /= 0.0) d(:)=d(:)/abs(y(:))
rofunc=sum(x(:)*sign(1.0_sp,d(:)), mask=(abs(d(:)) > EPS) )

```

Sample page from NUMERICAL RECIPES IN FORTRAN 90: THE Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 by Numerical Recipes Software.  
Permission is granted for Internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-readable files (including this one), to any server computer, is strictly prohibited. To order Numerical Recipes books or CDROMs, visit website <http://www.nr.com> or call 1-800-872-7423 (North America only), or send email to [directcustserv@cambridge.org](mailto:directcustserv@cambridge.org) (outside North America).

```
END FUNCTION rofunc
END SUBROUTINE medfit
```

**f90** The organization of this routine is similar to that of amoeba discussed on p. 1209. We want to keep the argument list of rofunc to a minimum, but we want to make clear what global variables it accesses and protect medfit\_private's name space. In the Fortran 77 version, we kept the only argument as b by passing the global variables in a common block. This required us to make copies of the arrays x and y. An alternative Fortran 90 implementation would be to use a module with pointers to the arguments of medfit like x and y that need to be passed to rofunc. We think the medfit\_private construction is simpler.

Sample page from NUMERICAL RECIPES IN FORTRAN 90: THE Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)

Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 by Numerical Recipes Software. Permission is granted for Internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-readable files (including this one), to any server computer, is strictly prohibited. To order Numerical Recipes books or CDROMs, visit website <http://www.nr.com> or call 1-800-872-7423 (North America only), or send email to [directcustserv@cambridge.org](mailto:directcustserv@cambridge.org) (outside North America).